Strengthening MDA by Drawing from the Living Systems Theory

Alain Wegmann

Laboratory of Systemic Modeling
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Tel. ++41 (0) 21 693 43 81
Fax ++41 (0) 21 693 47 01
alain.wegmann@epfl.ch

Otto Preiss[1]

Department of Information Technologies
ABB Switzerland Ltd
Corporate Research
CH-5405 Baden -Dättwil, Switzerland

Tel. ++41 (0) 58 586 80 69
Fax ++41 (0) 58 586 73 65
otto.preiss@ch.abb.com

Abstract

OMG's Model Driven Architecture initiative comes at a time where information system builders and integrators have realized that application design and thus application interoperability is not primarily a technology issue but is about understanding different types of systems, involving different professional and scientific disciplines. Understanding requires thinking and thinking is modeling. Hence, MDA provides a framework for modeling systems, especially those systems that are relevant in the context of IT-system integration within and across company boundaries. However, model correspondence, i.e., relations and transformations among models and views in a consistent way, presents a challenging problem for MDA.

Bridging of disciplines was also a key motivation for the systems science communities, and in particular for the life sciences. A widely accepted theory about all living systems was developed by J. G. Miller – the Living Systems Theory. His theory is striking because the basic concepts and principles are applicable at all levels, i.e., for all types of livings systems, from a cell to a supranational organization. The Living Systems Theory thus provides a good basis for consistently relating different systems and different views. In this paper, we will show how the living systems theory can be used to go about the problem of model correspondences. In particular, we suggest that MDA explicitly use the notions of a model reality with organizational levels based on a modeling ontology that is derived from the living systems theory.

1 Introduction

Integration and interoperability has always been at the heart of software engineering for anything else than non-trivial software products and systems. Philosophically speaking, thinking is modeling [1] and thus, agreed upon or compatible models are key to "inter-think", i.e., to facilitate communication among humans, among computers, and among humans and computers. Approaching this line of thought from the other side, we can say that making things interoperable requires thinking. Since thinking is modeling we infer that we cannot make things interoperable without models. Hence, anything that tries to make explicit what we need to do anyway is a valuable contribution, even more so if it not only eases the cognitive processing of a group of humans but also addresses the ease of processing by computers, i.e., if it promises a concrete path to code generation and is not condemned to (overhead) documentation only [2]. The Model Driven Architecture is an attempt to tackle both.

It always was the mission of the Object Management Group (OMG) to alleviate the problems of IT integration and interoperability by means of vendor-independent specifications. The Model Driven Architecture (MDA) [3], being the next evolutionary step in that direction, provides a framework to tie the different standards together, so that the support for interoperability can be addressed with specifications that range over the entire life cycle of software products. For the practicing software engineer, this means raising the level of interoperability from a mainly syntactic interface level (realized through agreed upon IDLs) to a more expressive behavioral level (realized through agreed upon models).

Among the many challenges of MDA we believe that the issues circling around model correspondences and model mappings are the most fundamental and most important to be solved. These correspondences are not only important for abstraction/refinement relationships within a certain viewpoint but also for viewpoint correspondences. We believe that the notions of ***agreed system modeling ontology*** and ***agreed conceptualization***

[1] Corresponding author

are essential to establishing rigorous model correspondences. While the former refers to a basic set of reoccurring concepts and concept relationships in all the models, the latter refers to an explicit model of the reality that serves as the fulcrum point for model correspondences. In this paper we show how we can strengthen MDA in that we make it more generic by projecting onto it the perspectives of life science. Concretely speaking, we believe that we can contribute to solving correspondence issues by introducing an agreed modeling ontology and an agreed conceptualization based on Miller's Living Systems Theory [4].

In the subsequent sections 2 and 3 we introduce MDA and living systems theory, respectively. We then present how to interpret MDA through the eyes of living systems theory (section 4) and thus enrich MDA to facilitate model correspondences. Finally, we conclude in section 5 by restating our position and by emphasizing on its potential impact.

2 The Strategic Importance of MDA

In this section we briefly summarize MDA's key objectives and key features before we turn to the challenges faced by model correspondences. Readers familiar with MDA may skip the subsequent two sections and jump to the model correspondence challenges directly (section 2.3)[2].

MDA comes in a time where information system builders have realized that application design is not primarily a technology issue but about understanding, and hence modeling, different systems so that interoperability can be reasoned about on higher levels of abstraction. As Checkland puts it: "…whenever one system supports or serves another, it is a very basic principle of systems thinking that the necessary features of the system which serves can be worked out only on the basis of a *prior* account of the system being served. This must be so because the nature of the system served – the way *it* is thought about – will dictate what counts as 'service', and hence what functions the system which provides that service must contain." [6].

2.1 Goals of MDA

In line with OMG's vision of truly integrated computing environments and the many vendor-independent specifications they have been promoting since their introduction of the Common Object Request Broker Architecture (CORBA), OMG is now integrating these standards into a conceptual framework called MDA.

The main goals of MDA can be summarized as follows:

- Provision of a standardized approach to IT-based information system specification where the "what" and the "how" are separated, or as said in [3] "…that separates specification of system functionality from specification of the implementation of that functionality on a specific platform".
- Provision of guidelines for structuring specifications that are represented in models. This shall advance the art, science, and scope of information system modeling.
- The previous two goals shall enable integration and interoperability as well as system evolution in the face of ever changing platform technologies.

If MDA is widely adopted by the IT community, users of the MDA benefit since it [7, 8]

- Integrates what you've built, you are building, with what you will build in future.
- Remains flexible in face of constant changing infrastructure.
- Lengthens the usable lifetime of your software, lowering maintenance costs and rising return on investment.

It is our interpretation that the implicit and generalized goal of MDA is to strive for a unification of system modeling[3], where system refers to all those systems that are relevant to properly design an integrated computing environment. Particular attention is paid to carefully separate, but explicitly relate, technology independent and technology dependent models. While the current focus is on IT-based information system interoperability, this would also allow raising the level to business interoperability, i.e., glue together business processes across organizations.

[2] A collection of resources on MDA can be found in [5].

[3] MDA claims to understand systems in the system-theoretic sense [3].

2.2 Basic Concepts and Principles of MDA

MDA's premise is that system specifications are expressed as models. However, MDA defines explicitly neither what a system is nor what a specification is. Specification and model are used somewhat interchangeably because an implicit assumption seems to be that a model is a formal specification. The defined concepts start with the definition of the concept of a model.

In the following we list the major concepts and their informal definition according to the MDA draft specification [3].

- ***Model*:** A representation of a part of the function, structure and/or behavior of a system. A formal specification is a model. Formal refers to a precisely defined syntax and semantics. The latter in turn is defined in terms of real-world things or by means of translating higher-level constructs into other constructs that have a well-defined meaning.
- ***View*:** A model that is based on specific abstraction criteria. Note that there is no particular distinction to the term viewpoint.
- ***(Model) Mapping*:** The set of rules and techniques used to modify one model in order to get another model. Mapping is used for model transformation. In particular, it shall allow the transformations among and between PIMs and PSMs.
- ***Abstraction*:** Suppression of irrelevant detail. What is and is not in a model should be determined by the relevant abstraction criteria. One highlighted use of abstraction is zooming in and out of objects and object interactions.
- ***Refinement*:** A relation between two models - an *abstraction* and *realization* – where one is more abstract than the other, respectively. The refinement relationship is also expressed in a model. Decomposition and composition is a special type of refinement.
- ***Platform-Independent Model (PIM)*:** A formal specification (i.e., a model) of the structure and function of a system that abstracts away technical detail.
- ***Platform-Specific Model (PSM)*:** A formal specification expressed in terms of the specification model of the target platform. Note that this is still implementation language environment independent.

The underlying premise is that UML [9], which is defined in terms of the Meta Object Facility (MOF), is used as the formal core that defines what model elements can be used in an MDA compliant model and also how separate views, levels of abstractions, and refinements are to be represented.

2.3 The Correspondence Challenges of MDA

As mentioned above, MDA introduces, among others, the concepts of a model, views, and abstraction/refinement. It uses the UML model element *package* to separate views and levels of abstraction. A package defines a grouping of model elements. Its elements somehow suffice to represent an organized description of some state of affairs of a system. The interrelationship between any two models, i.e., their integration, is informally defined as a model correspondence. MDA also informally introduces the concept of model mapping, which we understand as a transformation relationship, i.e., as a means to express model correspondences. Naturally, these model correspondences, represented as package correspondences, are essential to model traceability and evolution of a coherent set of models. However, while the MDA acknowledges these important issues and discussed some occurrences of correspondences and mappings at a conceptual level, it does not provide concrete guidelines that would help to formalize mappings.

It is our position that both a ***common core of concepts* (a system modeling ontology)** and an ***agreed model of the reality*** (cf. Tarski's notion of agreed conceptualization [10]) greatly improve the definition of model correspondences. Other authors also acknowledge that one core of common concepts (object, action, etc.) exists on all levels [11] and that "…model transformation lies at the center of the MDA" [2].

However, an explicit and agreed upon model reality is not emphasized. This is partly because MDA relies on UML, whose analysis also revealed a lack of this explicit relation to a model reality [12]. Nevertheless, MDA does not prohibit such a reality model. To the contrary, it encourages generic model mappings. Thus, introducing a model reality and relating individual models to such a model reality can be seen as a generic, albeit indirect, means to map models.

3 The Living System Theory

In 1995 James Miller published his second edition [4] (in 1978 his first version) of a thorough cross-discipline analysis and synthesis of the functions and behavior of living systems. It is called Living Systems Theory (LST) and is a general theory about how all living systems "work" - from the individual cell to supranational organizations (such as the UNO). Because the LST is a general theory, all concepts are metaphorical, i.e., they are meant to be algebraically translated to the particular living system in systemic inquiry.

By definition, living systems are open and thus constantly interact with their environment by means of information and matter-energy exchanges. Miller's thesis is that (a) every living system has the same 19 critical subsystems, which carry out the 19 essential processes of every living system, and (b) every living system belongs to one of eight defined levels and is composed of components that themselves are systems on the next lower level.[4]

3.1 Goals of LST

The goals of LST are to unify the scientific and often discipline-specific approaches to study and model living systems. In particular, LST takes an integral approach to structural and behavioral sciences while still leveraging from individual disciplines. It applies the same general theory (i.e., it uses the same concepts and principles) recursively at all levels of complex living systems.

3.2 Basic Concepts and Principles of LST

The central concept of livings systems is that of a ***system***. Miller defines a system as follows:

"A system is a set of interacting units with relationships among them. The word "set" implies that the units have some common properties. These common properties are essential if the units are to interact or have relationships. The state of each unit is constrained by, conditioned by, or dependent on the state of other units. The units are coupled." [4]

The second most important concept is that of an ***organizational level***. According to Miller "…the universe contains a hierarchy of systems, each more advanced or 'higher' level made of systems of the lower levels". He identifies eight distinct levels for living systems. This distinction is tightly linked to our experience in perceiving and studying the world of living systems. It holds that these levels are practical (but not necessarily optimal) to describe the reality with enough simplicity. It is perfectly conceivable to define other levels. Miller chose these levels as they allowed him to do an extensive literature review related to these individual levels[5], resulting in the suggestion on how systems on all these levels could be modeled using a common meta-model.

According to the way we perceive natural systems, Miller's basic model of living systems introduces two principle views of a living system:

(a) ***A "physical" view***: A system is made of components. Components in turn are systems. The higher-level system, of which a system is a component of, is called supra-system. For example, an organ is the supra-system of several cells. As mentioned above, Miller defines eight levels of systems for living systems. These organizational levels are characterized by the fact that each level has the same types of components but different specializations. Systems at higher levels are more complex than systems at lower levels.

(b) ***A conceptual or "process" view***: A system has subsystems. Subsystems carry out the essential processes of the system. Subsystems interact to achieve a desired overall matter/energy or information processing. Subsystems of a system are always realized by the collaboration of one or more components. The notion of an echelon is used to refer to a hierarchical organization of components within a subsystem taking over certain types of activities. Miller's thesis is that every living system has the same 19 critical subsystems, which make out the "living" part of the LST. The defined subsystems process information, matter/energy, or both. Generally speaking, subsystems serve to structure behavior.

From the above definition of views it is easy to infer that every system is a component, but not necessarily a subsystem of its supra-system. The properties (or behavior) of a system as a whole emerge out of the interaction of the components comprising the system.

---

[4] Note that depending on the age of a referenced document, one may find eight or nine levels of livings systems and 19 or 20 subsystems. The eight levels are: cell, organ, organism, group, organization, community, society, supranational system.

[5] Approx. 400 publications are analyzed for each level.

4 Interpreting MDA using LST

Even if MDA's primary focus is to enable the interoperability of IT applications, the ultimate goal of its designers is to manage information system related models that represent an enterprise in all its dimensions - or disciplines. This point is essential for understanding the parallel that can be made with Miller's Living System Theory. The goal of Miller was to define a framework allowing the unification of all disciplines involved in the study of living systems. The parallel between both goals is striking. This for two reasons: firstly, MDA and LST address modeling of hierarchical systems; secondly they both deal with living systems such as people and enterprises[6]. As presented in Section 3, the body of knowledge developed by Miller is important and well founded. Could this knowledge help to improve MDA? In this paper we will address this question and more specifically how LST can improve model correspondence in MDA. To explain this, we will present three points: the need of an agreed conceptualization and an agreed system modeling ontology, the concept of organizational level, and the concepts of physical view and process view.

4.1 The Need for an Agreed Conceptualization and a System Modeling Ontology

When modelers make models, they identify entities they perceive as existing in the reality and they represent these entities as model elements in the model. In certain cases, the very same entity can be represented as different model elements by different modelers or by one modeler in different views. Different entities can also be considered as corresponding to one model element. In other words, the mapping between the reality and the model is not trivial. Quite often people refer to this relation as an "abstraction". The term "abstraction" is also frequently used for the action of removing details within a model. We believe that both meanings are fundamentally different and need to be differentiated. In the context of MDA, it is not clear if the relationship between the reality and the model is considered at all (beyond the statement that "semantics might be defined …in terms of things observed in the world being observed" [5]) or if the term "abstraction" designates also this relationship.

When a modeler represents her reality, she is using her knowledge and her perception to decide what to model. In other words, what a model represents is tightly linked to the discipline and the experience of the modeler [6]. In modeling an enterprise, multiple disciplines are involved. To be able to relate the models made by the specialists of the different disciplines, it is essential that they agree on what entities they consider as relevant in the reality and how these entities are modeled in the different models. In other words, they need to agree on the "conceptualization" of their reality. The term of "conceptualization" comes from Tarski's theory of truth. According to Tarski, a fact is true for a group of people if there is an agreed conceptualization between all people that the fact is actually true. In other words, truth does not exist in the vacuum and can be considered (at least in the context of enterprise modeling) as a social agreement between all people involved in the same project. As a consequence we propose to name the relationship between the reality and the model "conceptualization" (and not "abstraction"). Our recommendation for MDA would be to define explicitly the "conceptualization" relationship.

Once professionals agree that they need to conceptualize what they perceive as reality, they need to agree on the model elements that they will use to represent their reality in the model. We call the definition of these model elements a system modeling ontology[7]. So, to be able to build models that would describe an enterprise in all its dimensions, we need one system modeling ontology, used by all disciplines. We propose to base our ontology on the one defined by Miller. Note that the use of one ontology for all disciplines does not prevent each discipline to keep their specificities. In Miller's theory, all levels have different theories although they all share the same ontology.

Figure 4-1 depicts the previous discussion graphically. Instead of introducing n:n model correspondences/mappings (Figure 4-1 part (a)) without ever agreeing on the "model of reality" (which corresponds to the agreed conceptualization of the perceived reality), we would prefer to use an agreed upon model of reality based on Miller's living systems theory and relate the specific models to that agreed conceptualization (Figure 4-1 part (b)). This would help us come closer to precisely stated 1:n correspondences.

[6] Miller considers enterprises in the "organization" level. "Organizations are systems with multiechelon deciders whose components and subsystems may be subsidiary organizations, groups, and (uncommonly) single persons". [4]

[7] In the field of computer science, an ontology is defined as the concepts and the relationships between these concepts.

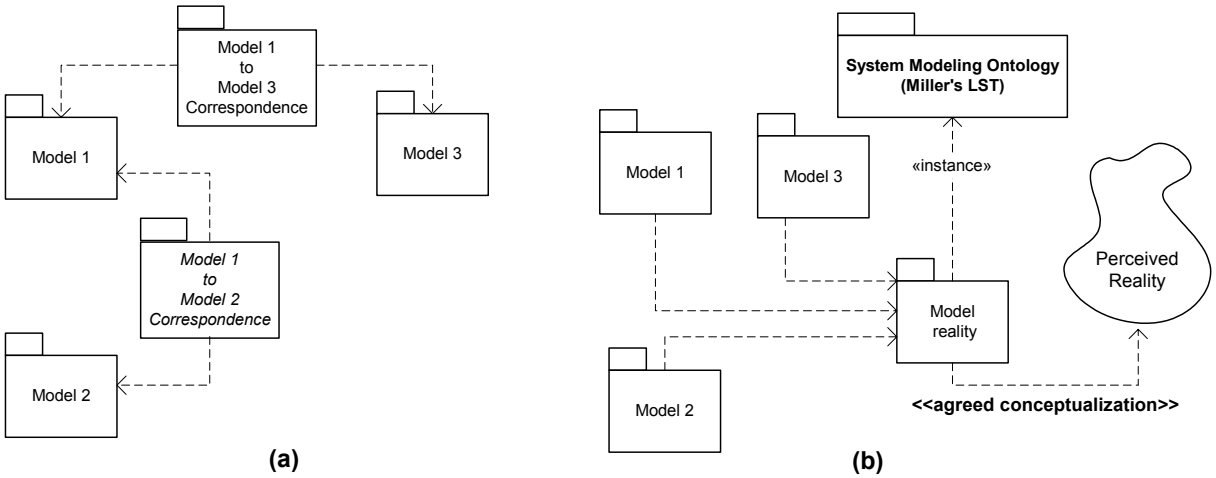

Figure 4-1: Model to model mapping (a) vs. model to model-reality mapping (b)

In summary, we recommend to differentiate between "conceptualization" and "abstraction" relationships and to consider the existence of an agreed "model of reality". Thanks to this, specialists of each discipline have the conceptual tool to discuss and agree about what exists in the reality. These recommendations do not have a profound impact on MDA. They simply propose a more detailed terminology to be able to address the problem of correspondence more effectively.

4.2 The Concept of Organizational Level

MDA proposes two models: platform independent (PIM) and platform specific models (PSM). A project might include multiple PIM and PSM as shown in the MDA meta-model ([3] Figure 8). This enables the description of an enterprise at multiple levels. For example, a PIM model might describe the business process in which an IT system is involved and a set of PIM/PSM might describe an IT implementation. Here again, the LST might provide a way to relate these models more explicitly.

When modeling an enterprise, modelers perceive the existence of organizational levels: business process, IT applications, software components, etc. These levels can even be quite detailed. E.g., in an IT-system we can perceive application servers, made of EJB components, which in turn are made of Java objects, made of byte code instructions. For example, in a project, in which we modeled the business and IT systems used to set prices and monitor purchases in a large, nation-wide, department store, we identified 12 levels going from the economical model down to the Java class model. This raises the question of the relation between PSM / PIM and the organizational levels. We believe that the "PIM to PSM to technology" mappings, as shown in Figure 4-2 (a), correspond to one dimension (which we call "technology dimension") and that the organizational levels correspond to an orthogonal dimension ("organizational level dimension").

The technology dimension represents the fact that a PIM model (describing what needs to be done) is transformed into a PSM model (describing how the platform will do it), which itself is mapped to the actual technologies. The organizational level represents the fact that we perceive the reality as made of hierarchical systems and that a specific analysis and design needs to be done for each level of the hierarchy. If we analyze Figure 4.2 (a), found in the OMG white paper from John Siegel [7], we notice that the PIM and PSM generate what is needed to deploy a web service. It is important to notice that there is no reference in the Figure 4.2 (a) to the business processes in which the IT system will be used. In addition, the source code of the software component providing the business logic in the web service is generated together with the configuration files of the web server. In Figure 4.2 (b), we have defined three organizational levels: the business level, the web service level and the software component level. In each level, we transform a PIM into a PSM that we map to the technology corresponding to the given level. In the business level, the referenced technology is the people and organizations. They are "programmed" by the definition of new or the modification of existing business processes. In the web service level, we generate the configuration files used to program the web server. In the software component level, we define the Java classes used to program the software component providing the business logic. Note that, in Figure 4.2 (b), we could also consider the generation of the configuration files and

the source code as being part of the same organizational level. This is a perfectly valid choice. In that case, we would have a model closer to Figure 4.2 (a). The point we want to illustrate with this example is not related to the number of organizational levels we should have but to the fact that these levels need to be explicitly defined and represented.

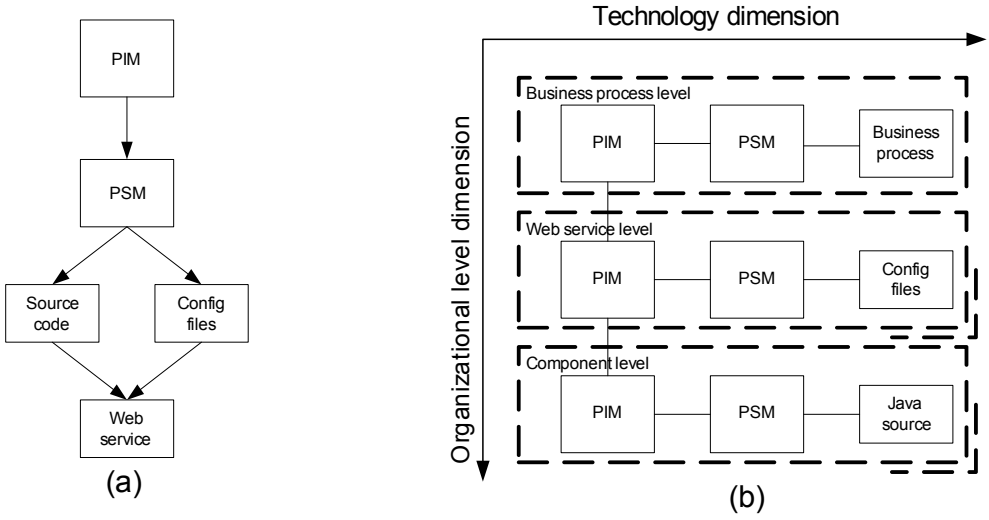

Figure 4-2: Example of PIM/PSM according to Siegel [7] and our interpretation of LST

In summary, our recommendation is that, within an MDA project, models are explicitly grouped by organizational levels. This grouping allows making explicit the correspondence between the models as the modelers have agreed on the hierarchy found in their perceived reality. Note that at each level, we will have PIM and PSM models. This corresponds to two orthogonal dimensions: the technology dimension and the organizational level dimension. This is somewhat analogous to the proposal made in KobrA, a recent software engineering method geared at component modeling [13].

4.3 The Concepts of Physical View and Conceptual (or Process) View

Miller presents the concepts of physical view and process view. How does it compare to MDA and what would it bring? The physical view matches what we have described in Section 4.2. In our example, the enterprise that implements a business process has web services and people working together to implement the process. In Miller's term, the web service and the people are called "components" of the system, which is the company. The physical view corresponds very well to the part-whole relationship that we know in software engineering.

The process view is more original. It defines conceptually the processes happening in the system of interest. In the process view, there are no references to the system implementation. It is a very useful view to represent the goal that needs to be achieved by the system of interest (regardless of its implementation). In order to analyze commonalities between systems, Miller proposes the concept of subsystem that corresponds to some kind of standard behavior (or pattern) that all systems implement. Examples of subsystems are: input transducer, decoder, timer, memory, decider, encoder, output transducers. It is interesting to highlight the parallel between these subsystems and the taxonomy of objects introduced by Jacobson [14]: interface objects, control objects, and entity objects. With these objects Jacobson has identified three kinds of subsystems (interface, control, and entity) and has automatically mapped them to what Miller would call components (objects in Jacobson's terminology). Applying Miller's concepts to Jacobson's approach, we would have a model describing the process in terms of interface, control and entities (subsystems in Miller's terms) but we would not have any reference to any objects (components in Miller's terms). It is only later in the development that we would map these three subsystems to one or more components.

In fact, the process view describes the behavior of the system of interest when the system is considered as atomic. The fact that components are abstracted explains why this view is a conceptual view. The process view corresponds to a level of abstraction which is higher than the one proposed by the PIM; but, as stated above, this level of abstraction is useful to specify the goal to be achieved by a system. Note that the process view of a system can be considered as the role of a system in the process in which it participates. In our MDA-LST

terminology, the process view of a system corresponds to the PIM model of the highest organizational level. This is an original contribution to UML and MDA, which was first discussed in [15]. Let us illustrate this contribution with the example of the web service. The web service is represented as an atomic system in the business process organizational level. What is represented is the business process and the roles of each system (people or IT) in the business process. The originality of our approach is to use the process view to specify the system's roles. This allows us to have the same set of models to define all organizational levels. This enabled us to develop a truly recursive design method that we call SEAM ("Systemic Enterprise Architecture Methodology") [16]. Our goal for SEAM is to fully, seamlessly, integrate business design and IT design.

In summary, we recommend to investigate how the concept of process view can be realized in MDA. The process view allows relating the model of a system (considered as atomic) in one organizational level to the model of the same system (considered as composite) in another organizational level. This improves the correspondence between models.

5 Conclusion

MDA is a good base not only for achieving interoperability of IT systems but also for unifying system modeling approaches. However, to get the full potential of MDA and make it as generic as possible, it is useful to draw from concepts defined in Miller's Living Systems Theory. In particular, we recommend considering the fact that modelers need to agree on how to conceptualize their perceived reality, and that their conceptualization should be formalized using one common system modeling ontology. We also claim that this ontology should include the concept of organizational level to group related models and should have a concept of process view to relate the models of the same system in different organizational levels. These suggestions do not change fundamentally the structure of MDA, they are rather suggestions on how to interpret the MDA models in order to be able to better express the correspondences among them.

References

[1] P. Valéry, *Cahiers*. vol. Tome 1. Paris: Gallimard (collection "La Pléiade"), 1975.

[2] J. Bézivin, "From Object Composition to Model Transformation with MDA," in *Proc. Technologies of Object-Oriented Languages and Systems (TOOLS-39)*, 2001.

[3] J. Miller and J. Mukerji, "Model Driven Architecture (MDA)," Object Management Group, Draft Specification ormsc/2001-07-01, July 9 2001.

[4] J. G. Miller, *Living Systems*. Colorado: University Press of Colorado, 1995.

[5] OMG. (2002, June 15). Model Driven Architecture. Object Management Group, Inc. [Online]. Available: http://www.omg.org/mda/

[6] P. Checkland and S. Holwell, *Information, Systems and Information Systems - making sense of the field*. Chichester, UK: John Wiley & Sons, 1998.

[7] J. Siegel, "Developing in OMG's Model-Driven Architecture," Object Management Group, Paper Revision 2.6, November 2001.

[8] R. Soley, "Model Driven Architecture," Object Management Group, White Paper Draft 3.2, November 27 2000.

[9] G. Booch, I. Jacobson, and J. Rumbaugh, "OMG Unified Modeling Language Specification, Version 1.30," Object Management Group, Specification, March 2000.

[10] A. Tarski, *Logic, Semantics, Metamathematics*: Oxford University Press, 1956.

[11] D. D'Souza, "Model-Driven Architecture and Integration," Kinetium, White Paper Version 1.1, March 2 2001.

[12] A. Naumenko and A. Wegmann, "A Metamodel for the Unified Modeling Language: Critical Analysis and Solution," Swiss Federal Institute of Technology, Lausanne, Technical Report EPFL-IC/2002/0112002.

[13] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wust, and J. Zettel, *Component-Based Product Line Engineering with UML*, 1st ed: Addison-Wesley, 2001.

[14] Ivar Jacobson, M. Ericsson, and A. Jacobson, *The Object Advantage: Business Process Reengineering With Object Technology*, 1st ed: Addison-Wesley, 1995.

[15] A. Wegmann and G. Genilloud, "The Roles of "Roles" in Use Case Diagrams," in *Proc. Third International Conference on the Unified Modeling Language (UML2000)*, 2000.

[16] A. Wegmann. (2002, March 15). Systemic Enterprise Architecture Methodology (SEAM). [Online]. Available: http://lamswww.epfl.ch